1/PRTS

10/533772

JC17 Rec'd PCT/PTO 04 MAY 2005

Microcircuit card comprising means for publication its data objects

The present invention relates to a microcircuit card that can be inserted into a computer system, the card including a host computer system (processor, memories, etc.) and at least one data object (software programs, variables, files, etc.) used by the host system.

In the remainder of the present document, we will use:

- the term "terminal" to refer to the computer system, including a microcircuit card reader into which the card is inserted. The terminal may be for example, a personal computer (PC), a telecommunication device (telephone, personal digital assistant, etc.) or a digital data processing system designed to perform a particular kind of processing, for example cryptographic processing;
- the term "third-party system" to refer to a computer system connected to the terminal as defined above via a telecommunication network (local area network, Internet, etc.).

In known manner, the Java Card 2.2 standard defines an object language for simplifying the development of software applications executing on a microcircuit card. The host system of the microcircuit card conventionally includes an operating system (OS) etched into a read-only memory (ROM) of the card, this operating system including a Java Card virtual machine (JVM).

According to this standard, the host system of the card and the terminal communicate by means of a standard rudimentary communication protocol.

To be more precise, information exchanged between the host system and the terminal is coded in hexadecimal and placed in a table of bytes that constitutes either a command message from the terminal to the card or a response

message from the card to the terminal. A command or response message of this kind is known as an application protocol data unit (APDU).

Unfortunately, the Java Card 2.2 standard does not provide for publishing the list of the data objects accessible on the card.

This constraint makes it very difficult to develop software applications executing on a terminal system or a third-party system. This is because developing software applications executing on that kind of system necessitates fastidious coding of the APDUs, and in particular a knowledge of the hexadecimal codes for addressing the objects of the card. One example of a software application conforming to the Java Card 2.2 standard is set out by way of example in appendix A.

The present invention aims to solve the problems cited above. To be more precise, in accordance with a first aspect, the invention provides a microcircuit card including at least one data object, the card being characterized in that it includes:

- a register including a logical identifier of that object and at least one first reference of that object local to the card, and
- means adapted, on reception of a first message including the logical identifier, to communicate at least one second local reference obtained from the first local reference.

Correlatively, and according to a second aspect, the invention concerns computer equipment of terminal type including means adapted to implement a software application including at least one first instruction for using at least one data object in a microcircuit card, the first instruction using a logical identifier of that object. The terminal is characterized in that it includes means for obtaining, from the logical identifier, at least one second

local reference obtained by the microcircuit card from a first reference of the data object local to the card,

means for translating the first instruction into at least one second instruction that can be executed on said card, the second instruction using that second local reference, and communication means adapted to communicate the second instruction to the card for said use.

Thus a software application executing on the terminal can include instructions for executing a data object of the card and using a logical identifier of that object instead of a reference local to the card.

Moreover, the software application can include high-level instructions for exchange of information between the terminal and the host system of the card which are translated into APDU messages by the above translation means.

Thus the present invention facilitates the development of software applications executing on the terminal because the developer does not need to know the hexadecimal codes for addressing the objects of the microcircuit card or to code the APDU commands.

A data object of the microcircuit card can in particular be a computer program, a variable or a computer file.

According to one advantageous feature, the microcircuit card further includes means for publication of the logical identifier and of the first local reference in the register of the card.

In a first embodiment, the register of the card is created once and for all, i.e. in static manner at the time of creating the microcircuit card.

On the other hand, in a preferred embodiment of the present invention, the publication means enable dynamic publication of the logical identifier and the first local reference of the data object in the register of the card.

Thus only the data objects necessary at a given time are accessible to software applications executing on the terminal.

In a preferred embodiment, the data object is a Java Card type object belonging to a Java Card applet and the second local reference of the data object conforms to the Java Card standard. In this preferred embodiment, the obtaining means of the computer equipment terminal are adapted to obtain the second local reference conforming to the Java Card standard from the microcircuit card.

The publication is preferably performed at the initialization of that applet.

This preferred embodiment leaves it up to the developer of the Java applet executing on the host system of the microcircuit card whether to make each data object of that applet accessible or not to software applications executing on the terminal.

In a variant of the above preferred embodiment, the communication means are adapted to communicate an identifier of the applet on reception of the first message.

Thus in this variant the identifier of the applet is obtained without the programmer having to know its hexadecimal code. According to the Java Card standard, this identifier is needed if it is to be possible to use the data objects of the applet subsequently.

In another preferred variant embodiment of the microcircuit card of the invention, on reception of a second message, the communication means of the card communicate all the logical identifiers included in the register of the card.

This variant is particularly advantageous because it enables the terminal system to know all the data objects published by the microcircuit card and the various versions of those objects.

In a first variant embodiment, the second local

reference communicated by the communication means of the microcircuit card is the first local reference of the data object proper.

In a preferred variant embodiment, the second local reference is temporary and is obtained by encrypting the first local reference using an encryption key of the microcircuit card.

Thus the second local reference can be used only for a session defined by the selection of the applet. This preferred variant embodiment makes the card more secure, in accordance with the recommendations of the Java Card 2.2 standard, in order to guard against attack by "replaying", i.e. attacks aiming to make fraudulent use of the object of the card, by re-using its reference, without selecting the applet implementing that object.

In a preferred embodiment, the terminal further includes means for publication, in one of its registers, a buffer object including an interface identical to that of the data object of the card, that buffer object being adapted to translate an instruction executing on a third-party system and using the logical identifier into at least one second instruction that can be executed on the card and uses the second local reference.

This preferred embodiment enables the use of the data objects of the microcircuit card by a software application executing on a third-party system networked with the terminal of the card.

The register preferably conforms to the Java2 SE RMI Registry standard.

In a preferred variant of this embodiment, the publication means obtain and publish in this register of the terminal system all the buffer objects of the data objects published by said card.

This variant is particularly advantageous because it enables the third-party system to know all the data

objects published by the microcircuit card and the various versions of those objects.

The invention will be better understood and other advantages will become clearer in the light of the following description of a microcircuit card and a computer system terminal conforming to the principle thereof, the description being given by way of example only and with reference to the appended drawings, in which:

- figure 1 is a block diagram of a microcircuit card conforming to one particular embodiment of the present invention, and

- figure 2 is a block diagram of a computer system terminal conforming to a preferred embodiment of the invention.

Moreover, the description is accompanied by the following appendices:

- appendix A: source code of a Java Card applet of the figure 1 microcircuit card;

- appendix B: first part of the source code of a client code of the computer system terminal from figure 2;

- appendix C: syntax table of an APDU command for obtaining the identifier of an applet;

- appendix D: syntax table of an APDU command for obtaining the reference of an object;

- appendix E: syntax table of an APDU command for assigning a value to a data object in conformance with the ISO7816 standard;

- appendix F: second part of the source code of a client code of the figure 2 computer system terminal;

- appendix G: syntax table of an APDU command for executing a program remotely on a microcircuit card; and

- appendix H: syntax table of an APDU command for obtaining the logical identifiers of all the objects of a microcircuit card.

Figure 1 represents a microcircuit card 10

conforming to the present invention. The card 10 includes a host data processing system including a processor CPU associated in the conventional way with a read-only memory ROM, a volatile random-access memory RAM, and a non-volatile rewritable memory EEPROM.

The read-only memory ROM contains an operating system OS including a Java Card virtual machine JVM.

The microcircuit card 10 includes input-output means I/O adapted to send digital data to a terminal and to receive digital data from that terminal.

The input-output means I/O consist, for example, of a serial port known to the person skilled in the art.

The microcircuit card 10 described here contains three data objects.

As known to the person skilled in the art, an object conforming to the Java Card standard is a special case of a computer program. For simplicity, and by convention, in the remainder of this document an object may be regarded as a computer program.

Thus the microcircuit card 10 includes a first data object, namely a computer program belonging to a Java Card applet CalculatorApplet whose code (see appendix A) is stored in a rewritable memory EEPROM.

This data object includes a first reference 0060H local to the card 10 and conforming to the Java Card standard. In the prior art, the programmer of a software application executing on the terminal has to use this reference 0060H to execute this program on the card.

According to the present invention, computer program also includes a logical identifier "myCalculator" that can be used by a programmer of the computer system terminal in place of the reference 0060H.

The microcircuit card 10 contains a second data object, namely a file stored in the non-volatile rewritable memory EEPROM.

Note first that the term "file" as used in the present document must be understood in a broad sense, designating either a file or a directory. In particular, the term "file" is used to designate a directory DF (Dedicated File) and a file EF (Elementary File) in the sense of the ISO7816 standard.

As known to the person skilled in the art of microcircuit cards, the file includes, in accordance with the ISO7816 standard, a first reference local to the card known as the FID (File Identifier) and coded in hexadecimal on two bytes, in the present example "3F00H".

According to the present invention, this file comprises a logical identifier "CARD_HOLDER".

The microcircuit card 10 includes a third data object, namely a variable (or "object data") in a register of the rewritable memory EEPROM.

That variable includes a first reference local to the card coded in hexadecimal on two bytes, in the present example "0050H". It is assumed that this variable is used to store a date.

According to the present invention, this variable includes a logical identifier, here the character string "date".

According to the present invention, the microcircuit card contains a file 20 storing a table including one row for each data object accessible by a software application executing on a computer system terminal or on a third-party computer system.

Each row in the table in the file 20 includes:

- a record including the logical identifier of the data object, and

- a record including a first reference of that object local to the card.

The microcircuit card 10 includes an application CardManager stored in the read-only memory ROM, for

example.

In known manner, the application CardManager is adapted, using the input-output means I/O, to exchange messages consisting of APDU commands and responses with the computer system terminal into which the microcircuit card is inserted.

It is adapted in particular to receive from the above terminal a message Lookup_APDU including the logical identifier of a data object.

The application CardManager is also adapted, using the file 20, to communicate a second local reference of the data object to the terminal on reception of the message Lookup_APDU.

To be more precise, the application CardManager extracts the logical identifier contained in the APDU message, looks in the file 20 for the row containing that logical identifier in its second column, and obtains the first local reference contained in the second column of the same row.

In an embodiment described here, the first local reference obtained in this way is then encrypted using a function K that uses an encryption key KEY stored in the rewritable memory EEPROM, which makes it possible to obtain a second local reference communicated to the terminal by the application CardManager.

Alternatively, the second local reference communicated to the terminal is the first local reference itself.

In the preferred embodiment, the communication means are adapted, on reception of a message including the logical identifier of a data object conforming to the Java Card 2.2 standard, to communicate the identifier AID of the Java Card applet CalculatorApplet including that data object, for example the reference A000000000H.

Thus, on reception of a message including the

logical identifier "myCalculator", the communication means communicate the identifier AID A000000000H of the Java applet CalculatorApplet and the second local reference K(0060H)=0460h obtained by encrypting the local reference 0060H using the key KEY.

The application CardManager also supplies the name Calculator of the interface of the data object.

Moreover:

- on reception of a message including the logical identifier "CARD_HOLDER", the communication means communicate the second local reference K(0004H); and

- on reception of a message including the logical identifier "date", the communication means communicate the second local reference K(0050H).

In a first embodiment, the file 20 is constructed statically at the time of creating the card. For example, it is a table stored in the read-only memory ROM.

In the preferred embodiment described here, the file 20 is updated dynamically. To this end, the microcircuit card 10 includes means for publication the logical identifier and the first local reference in the file 20 of the card.

Here the publication means consist of a computer program Binding_API stored in read-only memory ROM and including a computer instruction bind, one example of the use of which, for publishing the program "myCalculator", is given in line A33 of appendix A.

As described here with reference to appendix A, this publication is preferably performed on initialization of the Java applet CalculatorApplet.

A computer equipment 100 of terminal type and conforming to the present invention is described next with reference to figure 2.

In the present example, the terminal 100 is a personal computer PC including means 110 for inserting a

microcircuit card 10 as described above with reference to figure 1.

The terminal 100 includes in particular means known in the art for implementing a software application, namely a processor CPU associated with ROM and RAM and a hard disk drive DISK.

The terminal 100 also includes input-output means I/O adapted to send digital data to and to receive data from a microcircuit card of the invention.

The input-output means I/O consist, for example, of a serial port known to the person skilled in the art.

According to the present invention, the terminal 100 can implement a software application DesktopApp stored on the hard disk drive DISK and including at least one first instruction for using a data object in a microcircuit card, that first instruction using the logical identifier of that object.

The code of the software application DesktopApp is set out in lines B21 through B31 of appendix B. That code includes in particular a line B23 using the logical identifier "myCalculator" of the computer program defined above with reference to figure 1.

In order to be able to use that logical identifier, and to ensure compatibility with the Java Card 2.2 standard, the computer equipment terminal 100 includes means for obtaining a second local reference from that logical identifier, through cooperation with the input-output means I/O. The second local reference is obtained by the microcircuit card 10 from a first reference of the data object local to the microcircuit card.

In the embodiment described here, the means for obtaining are provided by a program CardNaming.lookup stored on the hard disk drive DISK and whose source code is set out in lines B12 through B20 of appendix B.

In the embodiment described here, the means for

obtaining first obtain the identifier AID of the applet CalculatorApplet including the data object:

(B14) :byte[]AID= CardNaming.cardManagerLookup("myCalculator");

In the example described here, the method CardNaming.cardManagerLookup constructs a first APDU command containing the logical identifier myCalculator and sends that APDU command to the card.

The card responds by sending back the corresponding AID identifier, i.e. the value A000000000H.

This first APDU command causes execution on the card of the instruction Lookup_AID of the application CardManager which obtains from the file 20 the value of the AID associated with the logical identifier myCalculator.

In a known manner (after sending an APDU command for selecting the application CardManager, in accordance with the ISO7816 standard), the first APDU command is sent by the method CardNaming.cardManagerLookup using the class CardService predefined by the OCF (Open Card Framework) consortium for sending APDUs to a card. The syntax of this APDU command is set out in appendix C.

In a second stage, the program CardNaming.appletLookup sends a second APDU command with two input parameters, namely the identifier AID (A000000000H) of the applet obtained previously and the logical identifier myCalculator of the data object of the card.

The program CardNaming.appletLookup obtains in return the temporary second local reference K(0060H)=0460h obtained by encryption from the first local reference 0060H of that object on the microcircuit card.

(B15) byte[]ref= CardNaming.appletLookup(AID, "myCalculator");

This second APDU command, whose syntax is set out in appendix D, causes the execution on the card of the instruction lookup_reference of the application CardManager that obtains from the file 20 the first local reference

0060H associated with the logical identifier myCalculator and then calculates the second local reference 0460h.

The computer equipment terminal 100 further includes means for translating the first instruction into at least one second instruction that can be executed on said card, that second instruction using said at least one second local reference.

Thus for example, if the software application DesktopApp wishes to modify the content of the variable of the microcircuit card 10 whose logical identifier is "date" with the value "01/01/2002", the translation means generate a second instruction in the form of a command Put_data APDU, whose syntax is set out in appendix E. That instruction uses the first local reference 0050H of that object.

In the embodiment described here, the translation means consist of a proxy object created from the second local reference of the object (0460H) previously obtained. In this embodiment, the proxy object is an instance of the class Proxy known to the person skilled in the art and defined by the Java 2™ platform Standard, Edition V1.3.

According to the Java 2™ platform, Standard Edition V1.3, a class CardHandler is constructed that includes a method invoke for translating method calls in cooperation with the object proxy. The method invoke uses the class CardService defined by the OCF (Open Card Framework) consortium. The pseudocode of the class CardHandler is set out in lines B1 to B11 of appendix B.

In the example described here, the object proxy is constructed (lines B16 and B17) and returned (line B18) by the method CardNaming.lookup.

Accordingly, after execution of the instruction of line B23 of appendix B, an object calculator is created in the terminal and enables translation of instructions using the data object of the card whose logical reference is

"myCalculator".

The second reference ref local to the card of the translation object calculator is stored by the manufacturer of the class CardHandler described above (line B4, appendix B).

The translation object calculator may then be used in a program of the terminal to execute methods of the object of the microcircuit card whose logical reference is "myCalculator".

For example, the developer of this kind of program can write line B24 of appendix B to execute addition of the numbers 5 and 6 on the microcircuit card.

On execution of the instruction from the line B24, the method Invoke of the object CardHandler described above creates and sends the APDU command Invoke conforming to the Java 2.2 standard. The syntax of this command is set out in appendix G.

The applet CalculatorApplet has been selected beforehand, at the time of execution of the line B15.

According to the Java Card 2.2 standard, the instruction invoke communicated in this way to the applet CalculatorApplet by means of the APDU command initiates execution of the method add with the parameters 5 and 6.

The present invention further enables a software application executing on a third-party system to access a data object of the microcircuit card using its logical identifier when the card is inserted into a terminal connected to the third-party system via a telecommunication network.

To this end, and in a preferred embodiment, the computer equipment terminal 100 includes means for publication in a register Standard RMI Registry of a volatile random-access memory RAM of the terminal a buffer object remoteCalculator having an interface identical to that of the data object of the microcircuit card and

enabling an application executing on a third-party system to use that data object.

To this end, an intermediate object calculator_invoker is first created (line B25, appendix B) from the translation object calculator described above.

The object calculator_invoker belongs to the class InvokerImpl, which is itself a class UnicastRemoteObject known to the person skilled in the art and implementing the Java SE 1.3 RMI mechanisms. The code of the class InvokerImpl is set out in appendix F, lines F3 to F8.

In known manner, the intermediate object calculator_invoker includes the translation object calculator and a method invokeMethod, that method invokeMethod implementing a mechanism for translating calls to methods of the object calculator using the reflectivity mechanism of the Java language (line F7).

The object calculator_invoker created in line B25 is an RMI object stored in the register Standard RMI Registry of the terminal but does not have the methods of the data objects on the microcircuit card. In particular, it does not have the method add.

To enable use of that method by an application executing on a third-party system, there is stored in the register Standard RMI Registry of the terminal the buffer object remoteCalculator (lines B27 to B29) that implement the methods of the object of the card whose logical identifier is "myCalculator", the object remoteCalculator being also identified by the logical identifier "myCalculator" (line B29). The object remoteCalculator contains also the logical identifier and the RMI call means to the intermediate object calculator_invoker, as can be seen in the code of the class InvokeHandler, lines F9 to F15. The buffer object remoteCalculator is adapted, by calling on the object calculator_invoker, to translate an instruction that is executed on the third-party system and

using said logical identifier into at least one second instruction that can be executed on said card using the second local reference (0460H).

Thus two objects are stored in the register Standard RMI registry of the terminal, namely the intermediate object calculator_invoker under the identifier "myCalculator_Invoker" (line B26) and the buffer object remoteCalculator with the logical identifier "myCalculator" (line B29).

Accordingly, if a software application of the third-party system executes the following instruction:

```
Calculator calculator_third_party=(Calculator)Naming.lookup("myCalculator")
```

the Java 2 SE method Naming.lookup reads the buffer object remoteCalculator in the register Standard RMI registry of the terminal system, reconstructs it in the third-party system, and assigns it to the object calculator_third_party.

The software application of the third-party system can then use the object calculator_third-party to access the corresponding object of the microcircuit card 10 via the intermediate object calculator_invoker of the terminal system.

Thus the data objects of the microcircuit card 10 can be used on a third-party computer system networked to the computer system terminal 100.

In the preferred embodiment described here, the terminal system also supplies the publication means described above for the publication of the object calculator in the form of a class Java BindingService whose principal instructions, necessary for understanding the invention, are set out in appendix F, lines F16 to F30.

The publication means BindingService are in particular adapted to obtain and to publish in the register

standard RMI Registry of the terminal all the buffer objects of the data objects published by the microcircuit card.

In the preferred embodiment described here, the publication means BindingService use for this purpose the APDU command get_bound_objects_APDU whose syntax is set out in appendix G.

APPENDICES

APPENDIX A

```
public interface Calculator extends Remote {
   short add (short a, short b) throws RemoteException;
}

public class CalculatorImpl extends CardRemoteObject implements Calculator {
   public CalculatorImpl ( ) { super ( ); }
   short add(short a, short b) throws RemoteException {
      return (short) (a+b);
   }
}

public class CalculatorApplet extends Applet {
   private Dispatcher dispatcher;
   private final static byte CALCULATOR = { (byte)m, (byte)y, (byte)C, (byte)a, (byte)l
      , (byte)c, (byte)u, (byte)l, (byte)a, (byte)t, (byte)o, (byte)r };

   private CalculatorApplet{
   Calculator calculator = new CalculatorImpl ( );
   OCSRMIService rmi = new OCSRMIService (calculator) ;
   RMIRegistry.bind (CALCULATOR, (short) 0, (byte) CALCULATOR.length,
                     calculator) ;
   dispatcher = new Dispatcher(1);
   dispatcher.addService(rmi, Dispatcher.PROCESS_COMMAND);
   }

   public static void install(byte[] buffer, short offset, byte length) {
      (new CalculatorApplet( )).register( );
   } ;

   public void process (APDU apdu) {install(byte[] buffer, short offset, byte length) {
   dispatcher.process (apdu);
   }
}
```

APPENDIX B

```
B1    class CardHandler implements Invocation Handler {
B2       private byte[] ref;
B3       CardHandler(byte[]ref) {
B4          this.ref=ref;
B5       }
B6       public Object invoke (Object proxy, Method method, Object[]params) {
B7          /* Translation code of the call to the method "method" */
B8          /* Construction of an APDU, using ref to start execution of the method
            "method" on the card */
B9          /* Sending of the APDU to the card and recovery of the response returned by
            the method */
B10         }
B11   }
B12   class CardNaming {
B13      public static Remote lookup(String name) {
B14         byte aid[] = CardNaming.cardManagerLookup (name);
B15         byte ref[] = CardNaming.appletLookup(aid, name);
B16         Cardhandler ch = new Cardhandler(K(ref));
B17         Remote proxy = (Remote) Proxy.newProxyInstance(…, ch, …);
B18         return proxy;
B19      }
B20   }
B21   public class DesktopApp {
B22      void main (String [] args) {
B23         Calculator calculator = (Calculator) CardNaming.lookup ("myCalculator");
B24         short result = calculator.add((short) 5, (short 6);
B25         InvokerImpl calculator_invoker = new InvokerImpl(calculator);
B26         Naming.bind("myCalculator_invoker", calculator_invoker);
B27         InvokerHandler handler = new InvokerHandler("myCalculator_invoker");
B28         Remote remoteCalculator = Proxy.newProxyInstance(…, handler,…);
B29         Naming.bind("myCalculator", remoteCalculator);
B30      }
B31   }
```

APPENDIX C

AID Lookup_APDU		
Field name (ISO 7816)	Hexadecimal value	Description of the content of the field
CLA	80h	Proprietary call class
INS	42h	Instruction lookup_AID
P1	00h	No parameter
P2	00h	No parameter
Lc	0Ch	Length of the logical identifier **myCalculator**
Data field	6D7943616C63756C61746F72h	Logical identifier myCalculator coded in UTF-8 format
Le	00h	Expected size of the response unknown

APPENDIX D

Reference Lookup_APDU		
Field name (ISO 7816)	Hexadecimal value	Description of the content of the field
CLA	80h	Proprietary call class
INS	40h	Instruction lookup_reference
P1	00h	No parameter
P2	10h	Special parameter
Lc	0Ch	Length of the logical identifier
Data field	6D7943616C63756C61746F72h	Logical identifier coded in UTF-8 format
Le	00h	Expected size of the response unknown

APPENDIX E

Put_data APDU		
Field name (ISO 7816)	Hexadecimal value	Description of the content of the field
CLA	80h	Proprietary call class
INS	Dah	Instruction Put data
P1	00h	First byte of the tag date
P2	50h	Second byte of the tag date
Lc	04h	Length of date 01/01/2002
Data	01012002h	Date 01/01/2002

APPENDIX F

```
public interface Invoker extends Remote {
   public Object invoke (OMethod method, Object[] args);}

public class InvokerImpl extends UnicastRemoteObject implements Invoker {
   private Remote cardObject;
   InvokerImpl(Remote cardObject) {Super(); this.cardObject = cardObject;}
   public Object invokeMethod(String method, Object[] args) {
               /** use Reflectivity **/      }
}

public class InvokerHandler implements InvocationHandler, Serializable {
   private String invokerName;
   InvokerHandler(String invokerName) {this.invokerName = invokerName;}
   public Object invoke(Object proxy, Method method, Object[] args) {
      Invoker invoker = Naming.lookup(invokerName);
      return invoker.invokeMethod(getString(method), args); }
}

public class BindingService {
   public static main(String[] args) {
      String registryURL = args[1];
      Iterator iterator = getBoundObjects();
      while (iterator.hasNext()) {
         String name = (String)iterator.next();
         Remote cardObject = CardNaming.lookup(name);
         InvokerImpl invoker = new InvokerImpl(cardObject);
         String invokerName = registryURL + name + "_Invoker";
         Naming.bind(invokerName, invoker);
         InvokerHandler handler = new Invokerhandler(invokerName);
         Remote remoteObject = Proxy.nexProxyInstance(…, handler,…);
         Naming.bind(registryURL + name, remoteObject);
      }
   }
}
```

APPENDIX G

Invoke APDU		
Field name (ISO 7816)	Hexadecimal value	Description of the content of the field
CLA	80h	Proprietary call class
INS	38h	Instruction Invoke
P1	02h	First byte of version Java Card
P2	02h	Second byte of version Java Card
Lc	08h	Length of data
Data	0460 A7F6 0005 0006h	Data: reference (2 bytes), method add (2 bytes), parameter "5" (2 bytes) and parameter "6" (2 bytes)

APPENDIX H

get_bound_objects APDU		
Field name (ISO 7816)	Hexadecimal value	Description of the content of the field
CLA	80h	Proprietary call class
INS	44h	Instruction get_bound_objects
P1	00h	No parameter
P2	00h	No parameter
Le	00h	Expected size of the response